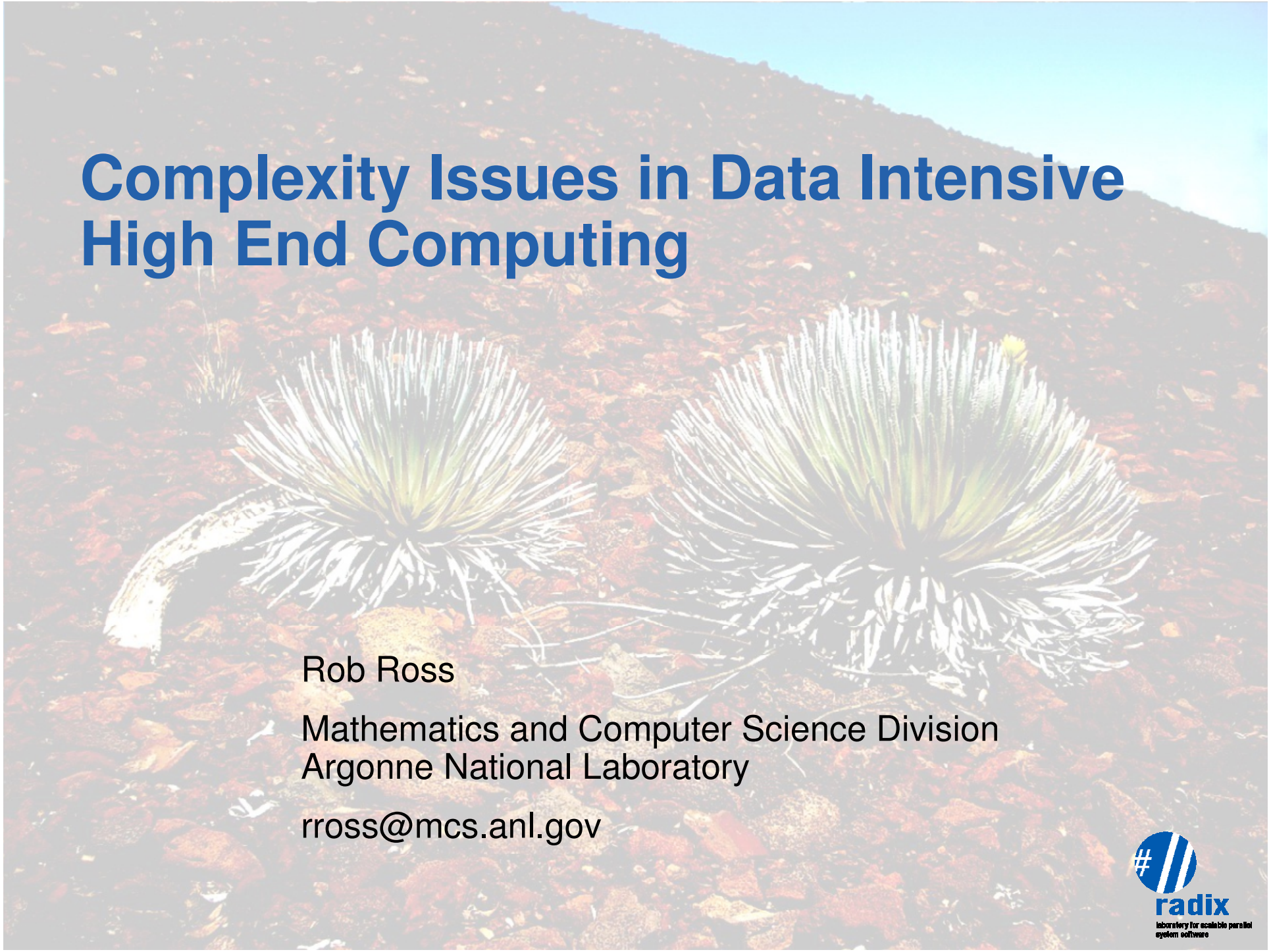
Complexity Issues in Data Intensive High End Computing



Rob Ross

Mathematics and Computer Science Division
Argonne National Laboratory

rross@mcs.anl.gov

radix
laboratory for scalable parallel system software

What's coming?

- Required I/O bandwidth is going up
  - Coordination (metadata) is the challenge
  - Tech. like NAND flash may mitigate this problem
- Spindle counts will go up to match bandwidth needs…
  - And network demands increase as well
- There will be more concurrent streams…
  - Unless we stop using POSIX (or POSIX-like) API
- The number of failure domains is increasing

New architectures and algorithms will be needed to meet these demands.

What have we done lately?

- Frangipani/Petal (1996) – virtual block devices
- Galley (1996) – subfiles (object-based storage)
- NFSv3 (1995) – best standard protocol to date
- POSIX I/O API (1996) – consistency semantics
- MPI-IO (1998) – standard parallel I/O API
- RAID (1988) – tolerating disk failures

We're mostly working with decade-old ideas.

radix
laboratory for scalable parallel system software

What can we do?

- Open up the design space
  - Give storage developers more options for solutions
- Leverage hierarchy in I/O systems
  - Hide some complexity from the storage system
- Build parallel storage solutions more quickly
  - It takes us 3-4 years to create a robust PFS
  - That's why we don't create truly novel solutions

radix
laboratory for scalable parallel system software

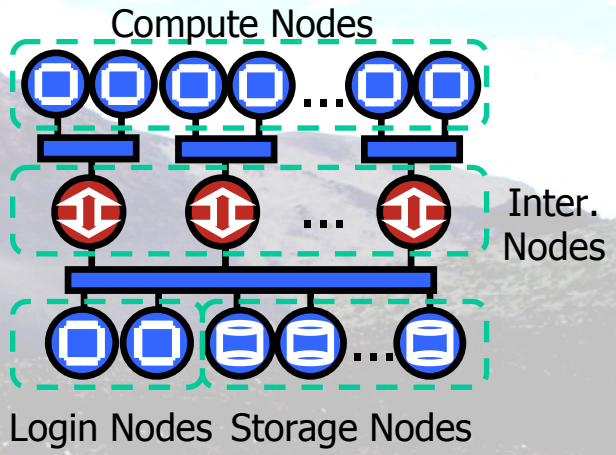
Opening up the design space

- Quit catering to users' whims
  - Give them better tools (APIs) instead
- POSIX is dead
  - Long live POSIX
- File-per-process is not a viable I/O model
  - We still spend time on this?
- Explore other organizations and consistency semantics
  - Can we learn from Google, physics data stores, peer-to-peer?

Figure 1: Possibly most popular tool for coding I/O inside parallel scientific applications

Leveraging hierarchy in the I/O system

- Partitioned systems becoming more common
- Intermediate nodes can aggregate operations
  - Much easier if application uses collective I/O
  - Result is fewer concurrent streams and more sequential access patterns



Common infrastructure for I/O forwarding could be shared across file systems.

radix
laboratory for scalable parallel system software

Building parallel I/O systems quickly

- We build most everything from scratch
  - CFS did well to use Portals as their transport
    - (Yes, Rob just complimented Lustre.)
  - Roger's example of rebuilding heartbeat/quorum
- **Need common components for system SW**
  - High-level messaging layer
  - Fault monitoring/reporting infrastructure
  - Object storage API for commodity disks

Summary

- Free developers to explore new designs
  - Wanted: Catchy name for enhanced POSIX
- Leverage the inherent hierarchy in HEC architectures to manage node counts
  - Build portable infrastructure to do this
- Develop and adopt common components to decrease time to solution

Real™ Open Source is critical.

radix
laboratory for scalable parallel system software